

CSE 512/CS 554
Homework Assignment 3 — Solutions

1. For a given problem dimension n , what is the optimal number of processors p for dense matrix-vector multiplication

(a) using a 1-D row or column algorithm on a 1-D processor mesh?

Answer:

$$T_p \approx \frac{t_c n^2}{p} + t_s p + t_w n,$$

so

$$T'_p \approx -\frac{t_c n^2}{p^2} + t_s = 0$$

when

$$p \approx n \sqrt{t_c/t_s}.$$

(b) using a 2-D algorithm on a 2-D processor mesh?

Answer:

$$T_p \approx \frac{t_c n^2}{p} + 2t_s \sqrt{p} + 2t_w n,$$

so

$$T'_p \approx -\frac{t_c n^2}{p^2} + \frac{t_s}{\sqrt{p}} = 0$$

when

$$p \approx \left(\frac{n^2 t_c}{t_s} \right)^{2/3}.$$

2. For LU factorization of an $n \times n$ matrix using a given number of processors p , what is the tradeoff point in the dimension n such that a 1-D row or column algorithm on a 1-D processor mesh is best for matrices of order less than n , and a 2-D algorithm on a 2-D mesh is best for matrices of order greater than n ?

Answer:

$$T_p^{1D} \approx \frac{t_c n^3}{3p} + t_s n + \frac{t_w n^2}{2},$$

and

$$T_p^{2D} \approx \frac{t_c n^3}{3p} + 2t_s n + \frac{t_w n^2}{\sqrt{p}}.$$

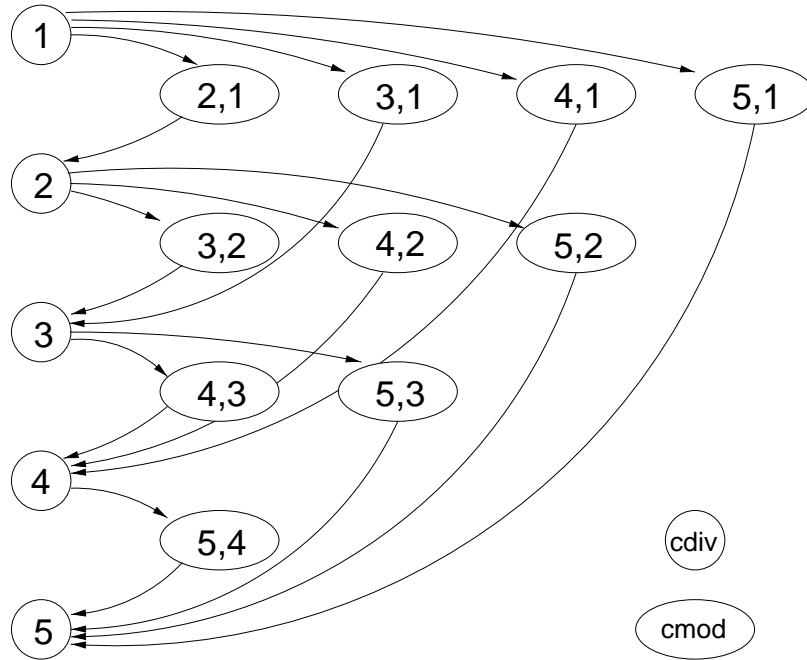
Setting these quantities equal, we see that the 2-D algorithm is better than the 1-D algorithm for

$$n > \frac{t_s}{t_w} \cdot \frac{2\sqrt{p}}{\sqrt{p}-2},$$

provided $p > 4$. If $p \leq 4$, then the 1-D algorithm is better than the 2-D algorithm for any value of n .

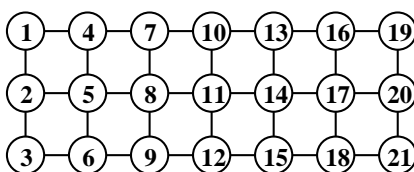
3. Using $cmod(j, k)$ and $cdiv(j)$ as tasks, draw a task graph (with tasks as nodes and data dependences as edges) for column-oriented Cholesky factorization of a dense symmetric positive definite matrix of order 5.

Answer:

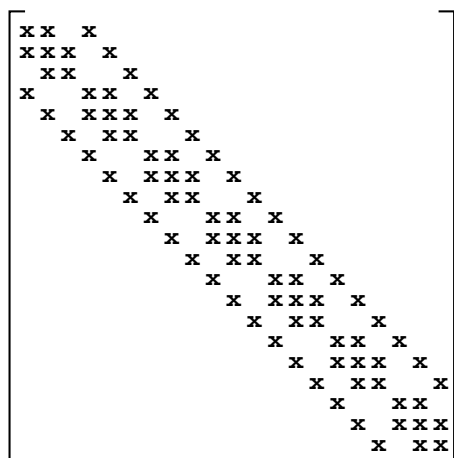


4. (a) Draw the sparsity pattern of a matrix of order 21 corresponding to a 3×7 rectangular grid numbered in the “natural” (say, column-wise) order, using \times for nonzero and blank for zero entries.
- (b) Draw the sparsity pattern of the Cholesky factor of the matrix, indicating original nonzeros by \times and fill by $+$.
- (c) Draw the elimination tree for the matrix.
- (d) What is the maximum number of *cdiv* operations that can be done simultaneously?

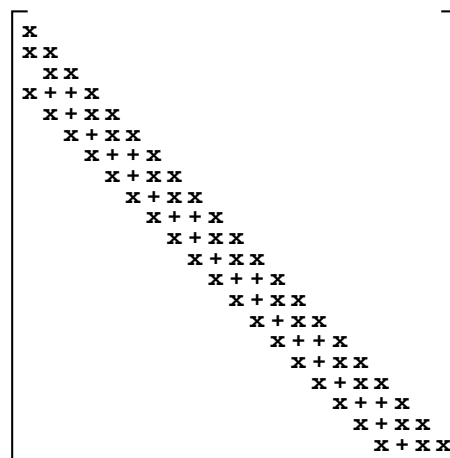
Answer:



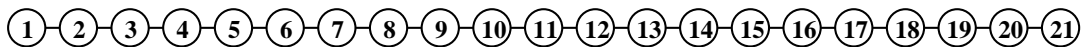
Grid graph



(a) Matrix sparsity pattern



(b) Cholesky factor

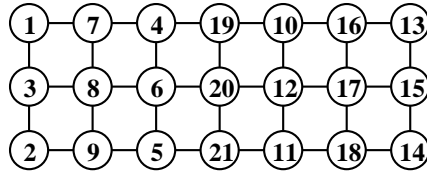


(c) Elimination tree

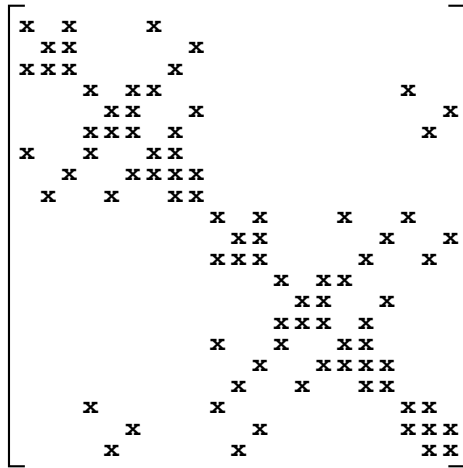
- (d) The elimination tree is a linear chain, so only one *cdiv* can be done at a time.

5. Repeat parts (a-d) of the previous problem using a nested dissection ordering.

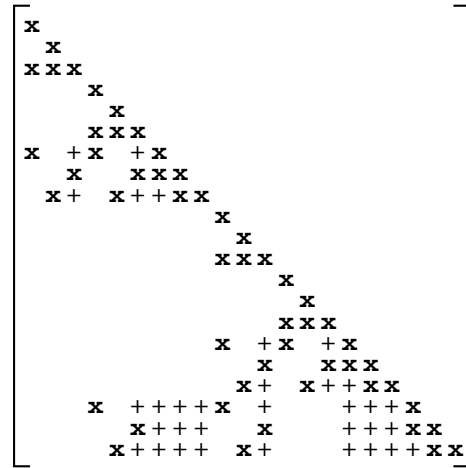
Answer:



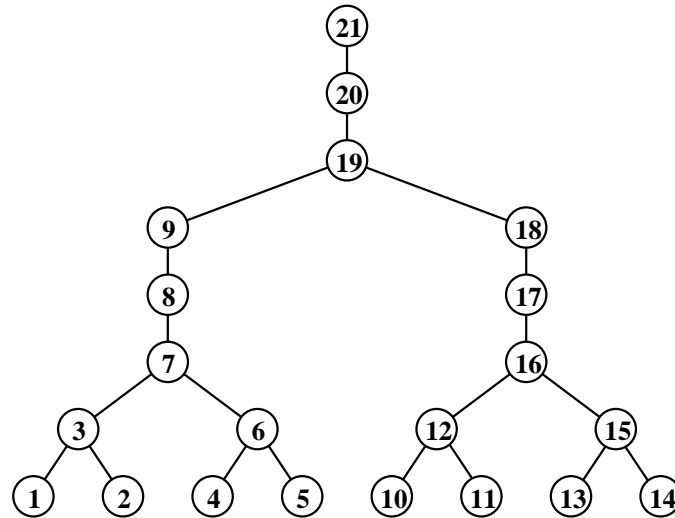
Reordered grid graph



(a) Matrix sparsity pattern



(b) Cholesky factor



(c) Elimination tree

(d) A maximum of eight *cdiv* operations can be done simultaneously, corresponding to the eight leaf nodes of the elimination tree.

6. (a) Formulate a 1-D row fan-out parallel algorithm for solving an *upper* triangular linear system $\mathbf{U}\mathbf{x} = \mathbf{b}$.

Answer:

```
for  $j = n$  to 1
  if  $j \in \text{myrows}$  then
     $x_j = b_j / u_{jj}$ 
  end
  broadcast  $x_j$ 
  for  $i \in \text{myrows}, i < j$ 
     $b_i = b_i - u_{ij}x_j$ 
  end
end
```

- (b) Formulate a 1-D column fan-in parallel algorithm for solving an *upper* triangular linear system $\mathbf{U}\mathbf{x} = \mathbf{b}$.

Answer:

```
for  $i = n$  to 1
   $t = 0$ 
  for  $j \in \text{mycols}, j > i$ 
     $t = t + u_{ij}x_j$ 
  end
  if  $i \in \text{mycols}$  then
    rcv sum reduction of  $t$ 
     $x_i = (b_i - t) / u_{ii}$ 
  else
    reduce  $t$  across tasks
  end
end
```

7. (a) In the 1-D column cyclic algorithm for solving triangular systems, how long does it take for the segment to return to a given processor, assuming that each intervening processor is ready to process the segment as soon as it arrives? Use the standard model for computation and communication.

Answer:

The segment, which is of size $p - 1$, must be sent sequentially p times, which takes a total communication time of $p(t_s + t_w(p - 1))$. Upon receiving the segment, each processor performs essentially p updates (one for each of the $p - 1$ elements of the incoming segment plus one new element), which takes time $p t_c$, where t_c is the time to compute one update. Such updating must be done sequentially by $p - 1$ processors, so the total computing time is $(p - 1) p t_c$. Thus, the total time required is

$$p(t_s + t_w(p - 1)) + (p - 1) p t_c = (t_w + t_c) p^2 + (t_s - t_c - 1) p.$$

- (b) How many component updates can be computed by the processor during this time?

Answer:

Dividing the above time by t_c , the time to compute one update, the number of components that can be updated during the time is about

$$\frac{(t_w + t_c) p^2 + (t_s - t_c - 1) p}{t_c}.$$